
Front matter

title: "Лабораторная работа 1" author: "Попова Юлия Дмитриевна, НФИМд-01-23"

Generic options

lang: ru-RU toc-title: "Содержание"

Bibliography

bibliography: bib/cite.bib csl: pandoc/csl/gost-r-7-0-5-2008-numeric.csl

Pdf output format

toc: true # Table of contents toc_depth: 2 lof: true # List of figures lot: true # List of tables fontsize: 12pt
linestretch: 1.5 papersize: a4 documentclass: scrreprt

l18n

polyglossia-lang: name: russian options: - spelling=modern - babelshorthands=true polyglossia-otherlangs:
name: english

Fonts

mainfont: PT Serif romanfont: PT Serif sansfont: PT Sans monofont: PT Mono mainfontoptions: Ligatures=TeX
romanfontoptions: Ligatures=TeX sansfontoptions: Ligatures=TeX,Scale=MatchLowercase monofontoptions:
Scale=MatchLowercase,Scale=0.9

Biblatex

biblatex: true biblio-style: "gost-numeric" biblatexoptions:

- parenttracker=true
- backend=biber
- hyperref=auto
- language=auto
- autolang=other*
- citestyle=gost-numeric

Misc options

indent: true header-includes:

- \linepenalty=10 # the penalty added to the badness of each line within a paragraph (no associated penalty node) Increasing the value makes tex try to have fewer lines in the paragraph.

- `\interlinepenalty=0` # value of the penalty (node) added after each line of a paragraph.
 - `\hyphenpenalty=50` # the penalty for line breaking at an automatically inserted hyphen
 - `\exhyphenpenalty=50` # the penalty for line breaking at an explicit hyphen
 - `\binoppenalty=700` # the penalty for breaking a line at a binary operator
 - `\relpenalty=500` # the penalty for breaking a line at a relation
 - `\clubpenalty=150` # extra penalty for breaking after first line of a paragraph
 - `\widowpenalty=150` # extra penalty for breaking before last line of a paragraph
 - `\displaywidowpenalty=50` # extra penalty for breaking before last line before a display math
 - `\brokenpenalty=100` # extra penalty for page breaking after a hyphenated line
 - `\predisplaypenalty=10000` # penalty for breaking before a display
 - `\postdisplaypenalty=0` # penalty for breaking after a display
 - `\floatingpenalty = 20000` # penalty for splitting an insertion (can only be split footnote in standard LaTeX)
 - `\raggedbottom` # or `\flushbottom`
 - `\usepackage{float}` # keep figures where there are in the text
 - `\floatplacement{figure}{H}` # keep figures where there are in the text
-

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра математического моделирования и
искусственного интеллекта

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

дисциплина: Математические основы защиты информации и
информационной безопасности

Преподаватель: Кулябов Дмитрий Сергеевич

Студент: Попова Юлия Дмитриевна

Группа: НФИмд-01-23

МОСКВА

2023 г.

Цель работы


Целью данной работы является приобретение практических навыков шифрования простой замены.[1]

Выполнение лабораторной работы

Требуется реализовать шифр Цезаря с произвольным ключом k и Реализовать шифр Атбаш.

Для этого я реализовал две программы на языке Python

Первая программа для шифра Цезаря(@fig:001)(@fig:002).

A screenshot of a code editor showing a Python script for a Caesar cipher. The script imports the sys module, defines the Russian alphabet as a string 'alpha', splits it into a list, and takes user input for a password and a shift key 'k'. It then calculates the unique letters in the alphabet not present in the password. Depending on the value of 'k', it either concatenates the unique letters to the password or rotates them. Finally, it prints the alphabet and the resulting cipher, and enters a loop to take a message to be encrypted, ending when the user enters 'f'.

```
import sys

alpha = "а б в г д е ё ж з и й к л м н о п р с т у ф х ц ч ш щ ъ ы ь э ю я"
alpha = alpha.split()
password = list(input("Пароль: ").lower())
k = int(input("Сдвиг: "))
k = k % len(alpha)

uniq_letters = list()
for letter in alpha:
    if letter not in password:
        uniq_letters.append(letter)
if k == 0:
    cypher = password + uniq_letters
elif k <= len(alpha) - len(password):
    cypher = uniq_letters[-k:] + password + uniq_letters[:len(uniq_letters)-k]

print(alpha)
print(cypher)

while True:
    mess = str(input("Предложение, которое будет зашифровано с помощью шифра Цезаря (f - для завершения работы программы): "))
    if mess == 'f':
        break
```

{#fig:001 width=100%}

```

print(alpha)
print(cypher)

while True:
    mess = str(input("Предложение, которое будет зашифровано с помощью шифра Цезаря (f - для завершения работы программы): "))
    if mess == 'f':
        break

    cypher_mess = str()
    for symbol in mess:
        if symbol == ' ':
            cypher_mess += ' '
        else:
            cypher_mess += cypher[alpha.index(symbol)]

    print(cypher_mess)

```

{#fig:002 width=100%}

Затем запустили программу, ввели пароль и сдвиг. Получили таблицу шифрования. Затем ввели предложение, которое нужно закодировать и получили зашифрованное сообщение. Вывод работы программы (@fig:003)

```

"C:\Users\Asuser\Desktop\магистратура\inf sec\lab1\venv\Scripts\python.exe" "C:\Users\Asuser\Desktop\магистратура\inf sec\lab1\main.py"
Пароль: пароль
Сдвиг: 3
['а', 'б', 'в', 'г', 'д', 'е', 'ё', 'ж', 'з', 'и', 'й', 'к', 'л', 'м', 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ъ', 'ы', 'ь', 'э', 'ю', 'я']
['а', 'ю', 'я', 'н', 'а', 'р', 'о', 'л', 'ь', 'б', 'в', 'г', 'д', 'е', 'ё', 'ж', 'з', 'и', 'й', 'к', 'м', 'н', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ъ', 'ы']
Предложение, которое будет зашифровано с помощью шифра Цезаря (f - для завершения работы программы): привет всем
аибярк яйре
Предложение, которое будет зашифровано с помощью шифра Цезаря (f - для завершения работы программы):

```

{#fig:003 width=100%}

Вторая программа для шифра Атбаш(@fig:004).

```

alpha = "а б в г д е ё ж з и й к л м н о п р с т у ф х ц ч щ ъ ы ь э ю я"
alpha = alpha.split()
alpha.append(' ')
cypher = alpha.copy()
cypher.reverse()

print(alpha)
print(cypher)

while True:
    finish = str(input("Предложение, которое будет зашифровано с помощью шифра Цезаря (f - для завершения работы программы): "))
    if finish == 'f':
        break

    cypher_mess = str()
    for symbol in finish:
        cypher_mess += cypher[alpha.index(symbol)]

    print(cypher_mess)

```

{#fig:004 width=100%}

Вывод работы программы (@fig:005)

```
"C:\Users\Asuser\Desktop\магистратура\inf sec\lab1\venv\Scripts\python.exe" "C:\Users\Asuser\Desktop\магистратура\inf sec\lab1\ata.py"
['a', 'б', 'в', 'г', 'д', 'е', 'ё', 'ж', 'з', 'и', 'й', 'к', 'л', 'м', 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ъ', 'ы', 'ь', 'э', 'ю', 'я', ' ' ]
[' ', 'я', 'ю', 'э', 'ы', 'ь', 'щ', 'ш', 'ч', 'ц', 'х', 'у', 'т', 'с', 'р', 'п', 'о', 'н', 'м', 'л', 'к', 'й', 'и', 'з', 'ж', 'ё', 'е', 'д', 'г', 'в', 'б', 'а']
Предложение, которое будет зашифровано с помощью шифра Цезаря (f - для завершения работы программы): привет всем
рпчмынаюоыу
Предложение, которое будет зашифровано с помощью шифра Цезаря (f - для завершения работы программы):
```

{#fig:005 width=100%}

Выводы

В результате выполнения работы освоили на практике шифрование простой замены. Шифр Цезаря и Атбаш.

Список литературы

- 1. Методические материалы курса